Infor Mongoose
Administrative User Guide

Important Notices

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

Trademark Acknowledgements

Table of Contents

Assigning User IDs and Passwords

Setting up users

Initially, only the supplied default system administrator user ID can create or delete other user IDs. User IDs and other information are set up on the **Users** form.

Setting passwords

Any user who is added to the System Administration group can access the **Users** form and change the password for any other user. Individual users can change their own passwords through the **User Information** form.

System administrators can use the **Password Parameters** form to specify password requirements, such as use of mixed case, numbers and special characters, and minimum/maximum password length. This form also allows you to set up rules for password expiration and locking out users after a certain number of password retries. For more information, see Setting Password Parameters.

Using token-based authentication

One option to increase system security is to require users to enter a second password—called a "passcode"—from a token-based authentication service. When this type of security is used, the **Sign In** dialog box displays a passcode field when the user attempts to log in. The user must obtain a passcode from a hardware or software "token" and enter it in this field to successfully log in.

For more information, see Token Authentication in Mongoose Applications.

Locked out users

If users enter an incorrect password more times than the specified number of retries (on the **Password Parameters** form), they will be locked out of the system for the number of minutes specified in Lockout Duration (on the **Password Parameters** form). If a user must log in to the system before the lockout time is up, a system administrator can manually override the lockout on the **Users** form by setting the User Login Status for the user to **Active**, which then resets the Login Failures to **0**.

Token Authentication in Mongoose Applications

One option to increase security and control access to your system is to use a token-based authentication service in tandem with the normal Mongoose login credentials. With this type of system, users must enter not only their user IDs and passwords, but also a second password, or *passcode,* obtained from a hardware or software "token".

To use a token-based authentication service in a Mongoose application, you must:

- Purchase access to the RSA SecurID token authentication service.

  At this time, the RSA SecurID service is the only token authentication service supported in Mongoose. You are responsible for purchasing and setting up this service for your system.
- Configure your Mongoose-based application to work with the token authentication service. This involves these basic procedures:
- Install and configure the Token Authenticator Service on the application (utility) server, using the Mongoose Configuration Wizard. This service is installed and run as an optional IIS service (but is required for token authentication of applications). This service can be used by many Mongoose-based applications simultaneously.

  For more information and the procedure, see the *Mongoose Installation Guide*.
- Configure the application in the Configuration Manager
- Set the passcode mapping for each user who will be using the service
- Let each user know how to use the token to sign in

Configuring the application in the Configuration Manager

To configure an application to use token-based authentication:

1. Open the Configuration Manager.
2. Select the **Application** tab.
3. From the list of available applications, select the application you want to configure.
4. Click **Edit**.
5. In the **Edit Application** dialog box, select the **Advanced** tab.
6. Select the **Token Authentication** option.
7. In the **Service URL** field provide the complete URL, in HTTP or HTTPS format, for the token authentication service.
8. (Optional but recommended) Click **Validate Service** to verify that the Mongoose system can make contact with the authentication service.

Setting the user passcode mapping

For users who will be required to use the token authentication service, you might need to provide a passcode mapping as part of the user profile: If the user's Mongoose user ID is the same as the RSA user ID for that user, then you need not provide passcode mapping; the system uses the Mongoose user ID. However, if the Mongoose user ID is different from the RSA user ID, then you must set the passcode mapping for that user.

To set the passcode mapping for a user:

1. Open the **Users** form and select the user.
2. Select the **Login Information** tab.
3. In the **Token Authentication Support, Passcode Mapping** field, specify the RSA user ID.
4. Save and close the **Users** form.

Letting users know how to sign in with token authentication

After you have configured the Mongoose system to use token authentication, you should let your users know:

- What they can expect when signing in—that is, that they will be required to enter a user ID, password, passcode, and configuration selection.

  NOTE: The **Passcode** login field displays only if the user selects a configuration that uses an application that is configured to use token-based authentication. This means that, when the user attempts to sign in, the **Passcode** field does not display until a configuration is selected that requires it.

- How the RSA SecurID token system that you have purchased works—that is, what kind of token they will use, how to obtain the passcode, and how to enter it.

Setting Password Parameters

Use these steps to specify parameters that apply to the password settings for users at all sites in an application database.

1 Open the **Password Parameters** form and specify this information:

Field or setting	Description/Values
Enforce Mixed Case	When this field is selected, user passwords must include at least one uppercase character and lowercase character, for example: MyPassword
Enforce Number	When this field is selected, user passwords must include at least one number, for example: mypassword1
Enforce Special Character	When this field is selected, user passwords must include at least one special character, for example: my#password Special characters include any character that is not either an alphabetic character nor a numeric digit.
Number of Retries	Specify the number of times users can re-enter their password before being locked out of the system. The default value is 3. A value of 0 disables the lockout feature for all users.
Lockout Duration (Minutes)	Specify the number of minutes users are locked out of the system following the maximum number of unsuccessful login attempts. The default lockout duration is 30 minutes.
Password Length Minimum	Specify the minimum number of characters required for user passwords. The default value is 7.
Password Length Maximum	Specify the maximum number of characters allowed for user passwords. The default value is 30, which is the most allowed by this application.
Password Expiration Days	Specify the number of days that are to elapse between a new password reset and its expiration. This calculation starts each time the user sets or resets the password.

	For example, if you specify a value of 60, and a user resets his password on June 1, the password expires on July 30. The Expiration Date on the Users form is set accordingly. If you reset this value, the change only affects users who modify their passwords after the change was made. The default value is 0, meaning the password does not expire.
Password Warning Days	Specify the number of days before the password expiration date that the user starts to see a warning that the password is soon to expire. The warning message displays each time the user logs in and includes the number of days left until the password expires. For example, if you specify a value of 5, and the user password is set to expire on July 30, the warning message starts to display on July 26. The default value is 0, which means that no warning is given. The value in this field must be less than the value in the Password Expiration Days field.
Password History Count	Specify the number of previous passwords that the system is to remember for each user. When a user creates a new password, it cannot match any other previous passwords that the system remembers.
Password Minimum Days	Specify the number of days that must pass after the user resets the password before the user can reset the password again. The intended use of this setting is to prevent users from quickly cycling through the Password History Count so as to effectively keep the same password all the time. The default value is 0, which indicates that there is no minimum number of days before the password can be reset. System administrators can override this value for users if necessary. For more information, see Password Set Date.

2 Save your changes.

The changes are applied to user accounts the next time they log in.

Assigning Users to Groups

You can set access levels and authorizations for individual users or for groups of users.

Group Authorizations

Default groups may already be set up; for example, assigning a user to an existing Purchasing group allows access to purchasing forms. To modify the forms and permissions for existing groups, use the Object Authorization for Group form. To add or delete groups, use the Groups form.

CAUTION: Although the system allows you to modify or delete default groups that are provided with your application, doing so may cause future conversion problems while upgrading and other problems. We recommend you copy the records from the default group to a new group name and modify that. Do NOT delete or modify default groups. For a list of the default Mongoose groups, see System Authorizations

User Authorizations

If a user is not assigned to any group, use the Object Authorizations for User form to determine what forms and privileges are available to that user.

How the Authorizations Work Together

Group authorizations allow you to control multiple users with one group. If an authorization is granted in one group and not granted in a second group, the least restrictive authorization is used.

For example:

- You can create a group COMaint that has EDIT and UPDATE privileges granted on the Customer Orders form. All other privileges on this form are not granted.
- You can create another group CO that has EXECUTE and READ privileges granted on the Customer Orders form. All other privileges on this form are not granted.
- Users in the COMaint group, who only have EDIT and UPDATE privileges, cannot open the Customer Orders form. Users in the CO group, who have EXECUTE and READ privileges, can open the Customer Orders form, but cannot make updates to it.
- "Power" users who are included in both the CO and COMaint groups can open the Customer Orders form and make updates.

Group authorizations work together. If a user is included in a group where a privilege is granted on a certain form, that granted privilege prevails over any "not granted" setting for the same form in other groups assigned to this user. However, any user authorizations set for individuals override group authorizations defined for a form.

At the User Authorizations level, privileges are either granted or revoked. There is only one set of privileges per form or per component per user. Therefore, if a privilege is revoked at the User Authorizations level, the same privilege cannot be granted at the Group Authorizations level.

User authorizations cannot have multiple privileges for the same form or same component. If a form or component is revoked at the User Authorization level, that revoke setting is used regardless of any group privileges that you specify.

If privileges are left blank at the user authorization level, the user is assigned the permissions defined at the group level.

User Authorization Report

In the **User Authorization Report**, user and group authorizations for forms and IDOs are grouped together by user ID. Row authorizations are grouped together by user ID and group name, and are sorted by IDO and group name. Options on the form let you choose the specific forms or IDOs you want to see in the report. You can see and compare all authorizations for a single user in the same section of the report. This makes it easier for you to determine whether a user has multiple permissions set differently for the same form, through different groups to which the user is assigned.

Example: Hiding Fields from Certain Users

You can control security at the component level. For example, you might allow outside salespeople or vendors to view some of your Mongoose forms through a URL, but you do not want them to see certain fields such as item costs. To do this for a group of users, use steps similar to these:

1. On the **Groups** form, create a group called, for example, Hide Costs - Outside Users.
2. Click **Group Authorizations** and add to this group the forms where you want to hide certain fields.
3. For forms where you want the outside users to be able to view but not update the forms, set the Read and Execute privileges to Granted and all of the others to Not Granted. For forms where you want to allow them to make updates, set all privileges to Granted.
4. Select one of the forms and click **Component Privileges** to set up privileges for specific fields (components).
5. In the **Name** drop-down list, select the component that you do not want the users to view, and set the Read and Update privileges to Not Granted. Leave the other columns blank.
6. Repeat steps 4 and 5 for all forms in the group.
7. On the **Users** form, select each of the outside users and assign them to this group.

Note: You can also set up component-level authorizations for individual users. On the **Users** form, click **User Authorizations** and follow similar steps.

Maintaining Application Schema Metadata

What Is Schema Metadata?

Schema metadata about tables includes:

- The application module in which a table belongs
- Data about table triggers, such as whether a primary key is generated, whether inserts or updates are allowed, and so on
- Data about AlphaKeys), which are system-generated values used as keys
- Data about user extended table (UET) inheritance
- Data about views generated for multi-site tables
- Data about application-specific table extensions, such as ERDBGW population for APS

The system refers to this metadata when generating table triggers.

NOTE: When you create a new table, first try generating and testing table triggers without adding your table to the metadata. If it works the way you want, do not add it to the metadata. Otherwise, refer to this section and the related topics to determine how to add metadata to adjust the generation.

Defining Metadata

After you create a custom table or a custom column on a standard table, define any needed metadata as follows:

1 Use the Application Schema Modules Metadata form to define any custom modules.

2 Use the Application Schema Tables Metadata form to:

- Assign custom tables to the appropriate module.
- For backward compatibility with existing multi-site tables, assign a viewname. The viewname should match the former table name. For example, a table formerly named item which is now item_mst would have a view named item. Specify a column name that contains the site value. The combination of view name and site value is used to create a view over the table.
- Specify information about insert and update triggers on the table.
- When UETs are added to the currently selected table, specify any other tables to which the UETs should be applied.

3 If a table has an AlphaKey (you want to generate Next Keys for a column in the specified table), use the Application Schema Columns Metadata form to define this.

If you later remove a custom table or column from the application database schema, you must also remove the corresponding row from the application schema metadata, if that row exists.

Generating Triggers

To generate triggers using the metadata you defined, use the Trigger Management form, specifying the tables where you changed metadata.

Triggers require generation after you:

- Add columns to standard tables.
- Add custom tables, if you want standard trigger functionality.
- Impact the schema for User Extended Tables with the UET Impact Schema form. (This generates triggers automatically and creates views for multi-site tables.)
- Upgrade databases where the schema has changed. (This generates triggers automatically.)
- Run the Database Migration utility. (This generates triggers automatically.)

Building and Running Scripts

You can dump the metadata into SQL scripts which can then be loaded into other application databases, for example at other sites where the same table/column changes are being made. Follow these steps:

1. In one site, define the metadata as described above.
2. Use the Generate Application Schema Metadata Scripts form to create the following scripts, as needed:
    - Module metadata, which was defined in the Application Schema Modules Metadata form
    - Tables metadata, which was defined in the Application Schem Tables Metadata form
    - Column (Next Keys) metadata, which was defined in the Application Schema Columns Metadata form.
3. Load the SQL script(s) into other sites (application databases), which must also contain any applicable new or modified schema elements.
4. At these other sites, use the Trigger Management form to generate triggers for specified tables.

Disabling Inactive Accounts

The system keeps track of the last date when any activity by each user in Mongoose was recorded. This information is stored for each user in the **Last Active Date** field on the **Users** form.

System administrators can set a value, in days, for the **User inactivity threshold** process default on the **Process Defaults** form. If a value is set, then when a user logs in, the user's **Last Active Date** is compared to the current date. If the difference is more than the number of days specified for the **User inactivity threshold**, the login is refused, and the account is disabled.

To enable a disabled account, change the user's status to **Active** on the **Users** form.

Maintaining or Discarding the Local Metadata Cache

When users log out of a session, the application caches metadata about IDOs and forms to two XML files on the local computer. The file can persist between sessions, so when users start a new session, the application uses the cached data, and users can open forms quickly.

The metadata cache files are:

- [config]IDOMetadataCache[windowsuser].xml
- [config]FormsMetadataCache[windowsuser].xml

These files are stored in [drive]\documents and settings\[windowsuser]\Local Settings\Application Data\Infor\WinStudio on the user's computer.

Discarding the Cache

The cache files contain the last login timestamp. If the IDO or forms metadata has changed since the last login, because of software patches or new customizations, the cache must be discarded. Caches are discarded when any of the following occur:

- The application checks the "last changed" timestamp for the IDO and forms data against the last login timestamp stored in the XML cache files. If the "last changed" timestamp is more recent than the last login timestamp, the application discards the appropriate cache.
- If a new user using a different language logs into the application using the same local computer, the forms strings cache on the local computer is cleared automatically.
- A user selects the **Form>Definition>Unload form and global objects** menu option to manually discard both caches on the local computer.
- A system administrator clicks the **Discard IDO Cache** button in the Configuration Manager utility for the specified configuration(s). This changes the "last changed" timestamp for IDO metadata, so the next time users connected to that configuration log in, their IDO metadata cache is cleared.
- A system administrator clicks the **Publish Form Change** button in the Configuration Manager utility for the specified configuration(s). This changes the "last changed" timestamp for forms metadata, so the next time users connected to that configuration log in, their forms metadata cache is cleared.

How "Last Changed" Timestamps are Updated

The objects database updates its "last changed" information via trigger, but the forms database does not.

Use the Configuration Manager utility's **Publish Form Change** button to reset the "last changed" timestamp when form changes have been applied via SQL rather than through the form server. For example, click this button when:

- You create new application event handlers that suspend inserts. You may need to manually mark forms metadata as modified, because the application caches forms information about whether new operations are suspended.
- You install an on-demand patch. (When you install a service pack, you do not need to click this button.)

Use the Configuration Manager utility's **Discard IDO Metadata** button to reset the "last changed" timestamp when changes have been made to IDO metadata, for example, when new IDOs have been added, or when new properties or methods have been added to existing IDOs.

Disabling Metadata Caching

Because the metadata cache could potentially consume a large amount of disk space on the local computer, system administrators can disable it by setting the **Persist WinStudio Metadata Cache** process default to **0** (zero).

Modifying Forms to Submit Tasks to the Background Queue

When developing a new form that includes a task that runs in the background, use these steps.

The application runs reports as background tasks that are placed on a "queue" to be run in order. You can also set up other tasks to run in the background. The TaskMan service polls the list of **Active Background Tasks** and executes any new task that is posted to the queue with a status of READY.

For more information about TaskMan, see the *Infor Mongoose Administration Guide* .

Setting up the Form

To set up the form:

1. Add a component called **BackgroundQueue** of type ObjMenuItem to the form. On the **Behavior** tab, add **RunBackgroundQueue** as the primary event.
2. Create a **RunBackgroundQueue** event of type **Run Form As Modal Child**. The parameters for the event should be as shown in the following example, where **GenerateReport** is the name of the form's **Run Background Task** event:

```
BackgroundQueue( SETVARVALUES(BGTaskName=V(BGTaskName),
RunTaskEvent=GenerateReport) )
```

3. Add the following parameters to the end of the parameter string for the **Run Background Task** event that submits the task:

```
TASKSTATUS(V(BGTaskStatus))TASKNUMBER(BGTaskNumber)
```

Use these keywords and parameters:

- TASKSTATUS is an optional input parameter to BGTaskSubmit. If the TASKSTATUS keyword is omitted, or if its value is anything other than WAITING, the task will be inserted into the ActiveBGTasks table with status READY and will be run by TaskMan. If TASKSTATUS is set to WAITING, the task will be entered in the ActiveBGTasks table with status WAITING. TaskMan ignores any records in this table with status other than READY or RUNNING.

- TASKNUMBER is an optional keyword used to specify a variable name that will hold the TaskNumber generated when a record is inserted into the ActiveBGTasks table.
- You can include substitution keywords that are replaced by appropriate values after the task is submitted to TaskMan.

Create a Background Task

1 Create a new record on the Background Task Definitions form.

2 Specify a task name (for example, RunCustomerOrderReport) and description.

3 Specify an executable:

- For a report, specify the name of the report (for example, CustomerOrder) in the first "executable" field and the type of executable (RPT) in the second field. The report definition (.RPT file) for this report must be placed in the TaskMan\Reports folder on the server where TaskMan and a report generating module are installed.
- For a stored procedure, specify the procedure name in the first field and select SP in the second field.
- For an executable program, specify the program name and path (for example, c:\Infor\myprog) in the first field and select EXE in the second field.

Either specify the complete path or use a path relative to the directory containing Infor TaskMan on the server. For example, assume you have an executable program called MyProg.exe that resides in mydirectory/mysubdirectory under the Infor TaskMan directory. In this case, you enter the following: mydirectory/mysubdirectory/MyProg

If you want TaskMan to handle the executable's connections to a database, the database must be identified to TaskMan through the TaskMan Configuration utility, and the executable must use the **B~** substitution keywords to connect. (See the WinStudio online help file for a list of substitution keywords.) Otherwise, it is up to the executable to handle all database connections.

- For an IDO method, specify the method name (in the format IDO.Method - for example, MyProgID.SLSites.MyTestMethod) in the Executable Name field and select IDOMTH in the Executable Type field.

For the task parameters, pass a comma-separated list of parameters that match the IDO method's parameters. You can either pass bare values (for example, "MyParameter1,MyParameter2") or use ~LIT~ syntax if whitespace is significant (for example, " MyParameter1,MyParameter2 ").

When processing tasks of type IDOMTH, TaskMan requires a configuration with the same name as the site name. On the machine where TaskMan is running (typically the utility server), you must create a configuration where the configuration name matches the database site name, if one doesn't already exist.

4 Click the buttons on the form to specify any report options or excluded tasks.

5 Save the record.

Your new background task can be called from any form as an event handler.

Check to See if the Task is Running

You can see which tasks are currently running by opening the Active Background Tasks form.

View Details About the Completed Task

Once TaskMan has completed the task, even if it fails, you can view details about its execution on the Background Task History form. Details include:

- Task description
- Return status
- User who submitted the task
- Messages triggered by the task

About the Multi-Lingual User Interface

Mongoose allows system administrators and individual users to select the language for the user interface. Language selection affects text in forms, messages, and help.

Culture Names and Folders

A language name is paired with a region or country name to form a culture name. For example:

Language - Country	Culture
English - United States	en-US
Spanish - Argentina	es-AR
Japanese - Japan	ja-JP

In Mongoose, culture names are used as names of subfolders under the working directory's folder. Each subfolder contains language-specific resources (which include the toolset's menus, dialogs, and strings, translated into the designated language) and may contain Help files translated into that language.

When the application starts, the toolset determines the current user-default settings, along with the settings in the **Language IDs** form, to display the user interface, Help, and messages in the language based on the settings.

If the language-specific resources are not available for your user-default settings, the user interface and/or Help displays in the application's base language (US English).

Language IDs

The Language IDs form defines the language-related IDs used in the application and links the IDs to resources. It specifies the strings table, language, and Help subdirectories assigned to each language ID, as well as the application message language used with the language ID and the typographic font in which reports print when the language ID is in effect. In addition, the form defines the date format and numeric format to use in report output when the language ID is in effect. Language ID names correspond to .NET culture names.

Translated Strings

The application's forms database includes a set of "strings" tables, which contain translations of the text strings used in form titles, field labels, buttons, and so on. The default strings table, for US English, is named Strings. Other strings tables have names like JapaneseStrings, FrenchStrings, and so on.

The strings table used for each language ID is determined by the settings in the application's Language IDs form. In the Sites form, the **Forms Database Name** should point to the forms database that contains the strings tables you want a specific site to use.

Translated Messages

Application message strings are maintained in the ApplicationMessages table. The table includes default text strings in U.S. English and any translations of message strings.

The language used for messages in an application is determined by the **Message Language** setting in the **Language IDs** form.

Overriding Language Selections

Users

Individual users can override the default language for an application in the **Settings** dialog box (on the **View** menu, select **Settings**.) In the **Language** box, you can select from a list of language IDs (which are defined in the **Language IDs** form). For example, a user-default language may be set to **French (France)[fr-FR]** because the user who usually sits at the computer speaks French. However, another user who speaks only English can override the language setting and view the user interface and Help in English by selecting, for example, **English (United Kingdom)[en-GB]**.

The language override information is saved and remains in effect the next time you log on to the application -- and until you reset the override. To reset the language to the user-default culture, select **<User Default Language Setting>** in **Language** in the **Settings** dialog box.

Administrators

The system administrator can modify the language assignments to create a mixed-language user interface. See the Language IDs form.

Limitations

- When overriding the language, if you select a language ID for which language-specific resources are not available, the interface will display in the application's base language (US English).

- Drop-down calendars always display in the language of the user-default culture.
- Graphs which are drawn by a third-party application may display text in the language of the user-default culture.
- Date and number formats in the application's user interface can only be changed through the Windows regional settings on the computer where the application's client resides. If you are using a remote desktop connection to log on to the system, this affects all users connecting to the application through that client.

Replication and Multi-Site

Example: How to Use the Site User Map Tab on the Sites Form

You have three sites that are sharing data through non-transactional replication: OHIO, ARIZ, and MICH. While logged into the OHIO site, set up Site User Maps for ARIZ and MICH, with the From Site set to OHIO and the user set to repl_user (this user is defined in all three sites). In ARIZ, set up Site User Maps connecting to OHIO and to MICH. In MICH, set up Site User Maps connecting to ARIZ and to OHIO.

In ARIZ, the Site User Maps setup looks like this for the three site records in the Sites form:

Site Record	From Site	User Name
ARIZ	(blank)	(blank)
MICH	ARIZ	repl_user
OHIO	ARIZ	repl_user

In OHIO, the Site User Maps setup looks like this for the three site records in the Sites form:

Site Record	From Site	User Name
ARIZ	OHIO	repl_user
MICH	OHIO	repl_user
OHIO	(blank)	(blank)

In MICH, the Site User Maps setup looks like this for the three site records in the Sites form:

Site Record	From Site	User Name
ARIZ	MICH	repl_user
MICH	(blank)	(blank)

OHIO	MICH	repl_user

Replication Steps

Prerequisites

Before you can replicate data to other sites, you must set up a multi-site environment.

You also should understand the basic concepts of replication: what data you need to replicate, and which sites you need to replicate the data to and from.

Set Up Intranets and Sites

NOTE: If you plan to set up shared tables and a master site for an intranet, use the steps in Setting Up Multi-Site Shared Tables when setting up that intranet and its sites, *INSTEAD OF* the rest of the steps in this topic.

1. Use the Intranets form to define the intranets that will group sites coexisting on a high speed network. For example, if you are running two sites and they are on the same LAN, define them in this application to use the same intranet so that you can use transactional replication at run time. Groups of databases may not be defined in the same intranet if they will replicate data but are not on the same version of the application.
2. Each site can have a system type associated with it. Use the System Types form to list the system types (application version number) you are using. For example, you might go to a higher version of the application for one of your sites, but still be using a lower version at another site.
3. Use the **Sites** form to define your sites and to determine the relationships between the sites. (You could also temporarily disable replication for a site from this form, if needed.)
4. You may want to use the Site Groups form to segregate your sites into different groups for reporting and cross-site data viewing. There is a default multi-site group that is used as the Site Group if you do not provide one. In that case, all sites are associated with the default group.

Set Up Replication Categories and Rules

Set up replication categories and rules for transferring data directly to or from application databases. The administrator for the target site can allow the source site to replicate some data to the target database, but prevent other data from being replicated. To do this, the administrator writes replication rules based on the replication categories, and then regenerates the triggers.

1. Use the Replication Categories form to specify the following and to group them into categories:
    - Tables that should be replicated.

Caution: If you try to replicate a table where RowPointer is part of the primary key, and you clear the **Retain Site** field for that table in the **Replication Categories** form, inserts would occur with a different RowPointer to the other site, so updates would never find that record. No failure would occur, but replication updates to the other site would simply not occur.

- Stored procedures that should be run at the target site.
- XML documents that should be sent to the target site.

The installation process creates some categories. Do not delete these standard categories. These categories have been created and tested to ensure that they handle the standard system processes. They should meet most of your needs without requiring any changes.

Caution: If you choose to create new categories or modify existing ones, we strongly recommend that you get help from Infor Professional Services. Determining all the relationships between tables and stored procedures is not a simple task.

2 Set Replication Rules for each site and category where replication should be performed. For example, if you want to replicate order entry information from Site A to Site B, create a replication rule (at both sites A and B) where the Source Site=A, Target Site=B, Category=Centralized Order Entry, and the Interval Type is either synchronous (Transactional) or asynchronous (sent to a queue at specified intervals).

Note: Replication to or from a site on an external intranet cannot be transactional. It must be non-transactional (asynchronous). If you set up a rule that violates this, an error message is displayed.

3 Use the button on the Replication Management form to regenerate the replication triggers. Any time you change a category or rule, you should regenerate the triggers.

Replicating Data to External Systems That Do Not Use Infor BOD Formatted XMLs

Suppose you want to integrate to another system that does not use the Infor BOD format for XML documents. As a system integrator you would take these steps:

1 Identify the application database tables for which changes need to be replicated to the other system.

2 Create an intranet for the other system (you need its URL).

3 Create a site for the other system, defining a system type, and adding the site to the intranet you created.

4 Create replication categories and replication rules that include the tables or transaction data that your system needs to push to the other system. Make sure your rules use an Interval Type that is *NOT* transactional.

5 Standard non-transactional replication automatically creates XML documents in a standard format that is specified by the IDO layer. However, unless the target application is another site

running the same application as this site, the other application probably will require and send its XML documents in a different format.

6 Use one of these options to handle the XML data:

- Build and install XSL transformation files to change the XML format produced by this application into a XML format the other system expects when importing the data. XSL transformation changes the data from one format to another in an XML document. For example, the external financials interface uses XSL transformations.
- Use the Replication Documents forms and the Replication Document Outbound Cross-References form to define or modify the metadata used to generate an outbound XML. Use the Replication Documents forms and the Replication Document Inbound Cross-References form to set up metadata to process inbound XMLs.

  XSL transformations just move the standard XML data into different formats, whereas modifying the metadata that generates the XML lets you request and send data that is not generated with the standard XML.

1 The replication system transmits the documents to the appropriate system.

For more information about replication, see the *Replication Reference* on our support web site.

Setting Up a Master Site and Shared Tables

About Shared Tables and Master Sites

In a multi-site environment where there are many sites, large amounts of data, and many users, you may want to set up one site as the master site for an intranet. In that case, certain _all tables and user tables can reside only on the master site database and are shared (read and written to through a SQL view) by other sites on the same intranet. No replication needs to occur for the shared _all and user tables, which can greatly improve system performance.

Steps to Set Up a Master Site and Shared Tables for All Sites on an Intranet

Follow these general steps. Refer to the forms and fields for detailed information. You can share _all tables, or user tables, or both types of tables.

NOTE: If you share _all tables or user tables, all sites in the same database must be on the same intranet.

Planning

CAUTION: You must plan your multi-site structure carefully before setting up the shared tables. This requires in-depth understanding of the SQL databases, this application's replication capabilities.

Refer to the *Multi-Site Planning* and *Replication Reference* documents on our support site for more information.

Set Up All Sites on the Intranet

The following steps assume that you have already used the Configuration Wizard during database server installation to link your multi-site databases.

1. In the **Intranets** form, specify all intranets to be used in this multi-site system. If you will be sharing _all tables, define an intranet that will include all the sites that share tables. This cannot be an "External" intranet. All the sites on this intranet must use the same version of this application.

   Do not define the master site yet - that will be done later.

2. In each site, use the **Sites** form to specify information about this site and the other sites that it relates to. Each site has a record in this form.
   - On the System Info tab, specify information about the site. Make sure the Intranet Name, Database Name, and Time Zone are set correctly for each of the sites listed on this form.
   - A list of linked sites displays automatically in the **Link Info** tab. The local site record shows links used in transactional replication between the currently selected site database and other site/entity databases. On site records other than the local site, the Link Info tab should show only links to the local site.
3. Use the Replication Categories form to specify tables, stored procedures, and XML documents that should be replicated and to group them into categories. The installation process creates some categories. Do not delete these standard categories. These categories have been created and tested to ensure that they handle the standard system processes. They should meet most of your needs without requiring any changes.
4. On the Replication Rules form, set up transactional rules for the **Site Admin** replication category between this site and all the other sites in the intranet, including the site that will be the master site. (Site Admin data includes tables such as site, intranet, and IntranetSharedTable.) You may also want to write other rules to replicate certain categories between certain sites. Even if a category contains _all tables that you are sharing, you probably want to write a replication rule for the category. (Any shared _all tables will not be replicated in this case.) Categories may contain additional base tables or stored procedures that are needed to perform certain functions. The rules you need should be determined in the multi-site planning phase.
5. On the Replication Management form, click **Regenerate Replication Triggers**. This ensures that site and intranet data is replicated to all the linked sites.

Set Up the Master Site

1. Log in to the site that you want to make the master site where the shared tables will exist for the intranet.
2. On the **Intranets** form, select this site's intranet. In the Master Site field, select this site to specify it as the master site for the intranet.

Share _All Tables

1 Log in to the master site.

2 On the **Intranet Shared Tables** form, select this site's intranet.

A list of the _all tables that can be shared displays. (Not every _all table is listed; some are not available for sharing.)

3 For tables that you want to be shared between all sites on the intranet, select **Shared**.

Another way to choose the shared tables is to select the replication categories that you want to share. When you select a Replication Category from the drop-down list and click **Select by Category**, the system marks _all tables in that category as Shared.

4 When you have selected all the tables you want to share, select **Actions > Save**.

5 Click **Process** to copy information from the tables at the other sites to the master site's table, delete the table from the other sites (creating views into the master site tables instead), and regenerate the replication triggers for the other sites. (When a site is in the same application database as the master site, the actual table is not dropped, because the master site table and the sharing site table are the same table.)

If you have selected several tables and have many sites on this intranet, processing may take a while. The **Processing Step** area displays the system's progress.

CAUTION: During processing, the selected tables are removed from all sites on the intranet except the master site. Unsharing (rebuilding the tables at the using sites) is time-consuming - so be very sure that you have everything set the way you want it before clicking the Process button.

6 During processing, the system validates link setup between the master site and the using sites of an intranet. If it finds a problem, an error message displays and nothing is processed; fix the link and then click Process again.

The **Processed** field indicates which tables have been processed - for example, if shared, they are now resident only in the master site's database. (Once a row on the form is marked as Processed, subsequent "Process" runs will not reprocess that row.)

After processing all the tables and sites, the system regenerates the replication triggers at the master site.

Share User Tables

Note: Intranet Licensing must be set up at the master site and all participating sites before you share user tables for the sites.

1 Log in to the master site.

2 On the Intranet Shared User Tables form, select this site's intranet.

Two lists of tables display:

- The top grid lists user tables that can be shared. You cannot edit this list. You can clear the Shared option for some tables, but not all of them, as described below.
- The bottom grid lists tables that contain a column whose base domain is UserNames.UserId or GroupNames.GroupId. The bottom grid is used during set up of shared user tables to

identify the tables and columns that may need to be updated if records that were formerly defined in the Usernames or Groupnames tables in the non-master site are moved to the master site, but with different UserId or GroupId values.

3 If you have custom tables that contain a column whose value comes from base domain UserNames.UserId or GroupNames.GroupId, add your custom table and its associated ID column to the Non-Shareable Tables grid. We recommend that your custom tables refer to the Username or Groupname columns, rather than the ID columns, because the distinct list of Usernames and Groupnames across intranet sites is always the same, whether they are stored in shared tables or per site, and therefore no changes are required for data referencing this base domain.

4 Select **Set up shared user tables**, which automatically selects **Shared** for all tables in the top grid and **Update Referenced ID** for all tables in the bottom grid.

5 In the top grid, clear the **Shared** check box for any tables that you do not want to share between sites.

Note: Be aware of how this works. For the Usernames, GroupNames, UserEmail, UserModules, UserPasswordHistory, UserCalendar, and UserTask tables:

- If you do not share these tables on the **Intranet Shared User Tables** form, these tables are:
  - Not shared across multiple sites, if one site per database
  - Shared if there are multiple sites per database
- If you do share these tables on the **Intranet Shared User Tables** form, these tables are:
  - Shared across multiple sites, if one site per database
  - Shared if there are multiple sites per database

For the AccountAuthorizations_mst and UserGroupMap_mst tables:

- If you do not share these tables on the **Intranet Shared User Tables** form, these tables are:
  - Not shared across multiple sites, if one site per database
  - Not shared if there are multiple sites per database
- If you do share these tables on the **Intranet Shared User Tables** form, then you can optionally share each of these tables across multiple sites, regardless of whether each site is in its own database or all sites are in the same database.

6 Click **Process** to copy information from the tables at the other sites to the master site's table, delete the table from the other sites (creating views into the master site tables instead), and regenerate the replication triggers for the other sites. (Removing tables and setting up views does not happen if the non-master site is in the same application database as the master site.)

If you have many sites on this intranet or many users and groups, processing may take a while. The **Processing Step** area displays the system's progress.

During processing, the system validates link setup between the master site and the using sites of an intranet. If it finds a problem, an error message displays and nothing is processed; fix the link and then click Process again.

The **Processed** field indicates which tables have been processed - for example, if shared, they are now resident only in the master site's database. (Once a row on the form is marked as Processed, subsequent "Process" runs will not reprocess that row.)

The **Status** field indicates whether the user tables are shared or not shared.

After processing all the tables and sites, the system regenerates the replication triggers at the master site.

7 After processing is complete, you must reapply a valid license document on the master site.

Adding a New Sharing Site (Changing the Intranet Value of a Site)

To add a new site to a existing shared tables intranet, see the appendix in the *Multi-Site Planning Guide* that describes how to add a new site to an existing intranet with shared tables.

Setting Up Shared User Tables at a New Site

When you are already sharing user tables, and you add a new site to the current master site's intranet, you may also want to set up the new site to share user tables. To do this:

1 In the Intranet Shared User Tables form, select the Set up per site user tables check box and click Change Setup Option to change the check box label to Set up shared user tables.

2 Click Process. The status of the shared tables is checked as each site is processed. If sharing has already been set up for a site, no processing occurs for that site, and the process continues with the next site. When it encounters a site that is not already set up, the site is processed.

Replication to Remote Sites

If there are other intranets with sites that want to replicate (*not* share) _all or user table data to/from sites in the sharing intranet:

- For tables that are shared, set up replication categories/rules between the master site and the sites on the other intranets.
- For tables that are not shared, set up replication categories/rules between any/all of the sites in the shared intranet and the sites on the other intranets.

Example

Intranet 1:

Site A (master site. Item_mst_all table is shared)

Site B

Intranet 2:

Site C

Site D

If Site B needs visibility into Site D's item data, replication rules should be set up from Site D to Site A.

If Site D needs visibility into Site B's item data, replication rules should be set up from Site A to Site D.

If Site C needs visibility into Site B's customer data (not a shared table), replication rules should be set up from Site B to Site C.

If a Shared _All Table Has Schema Changes

If one of the shared _all tables at the master site has a schema change, you will need to update the views into the _all table at the user sites on the intranet. To do this:

1. Log in to the master site and go to the **Replication Management** form.
2. Click the **Regenerate Views to Master Site** button.

Unsharing Multi-Site Shared Tables

About Shared Tables and Master Sites

In a multi-site environment where there are many sites, large amounts of data, and many users, you may want to set up one site as the master site for an intranet. In that case, certain _all and user tables can reside only on the master site database and are shared (read and written to through a SQL view) by other sites on the same intranet. For more information, see Setting Up Multi-Site Shared Tables.

Reverting Shared User Tables

If you need to revert shared user tables to reside again on the individual sites in the intranet, follow these steps:

1. Log in to the master site.
2. On the **Intranet Shared User Tables** form, select this site's intranet.

   Two lists of tables display:

   - The top grid lists the currently shared user tables. You cannot edit this list.
   - The bottom grid lists tables that contain a column whose base domain is UserNames.UserId or GroupNames.GroupId. When you set up per site user tables (that is, revert shared user

tables), this bottom grid is ignored during processing. All UserNames and GroupNames data are copied to the non-master sites as-is. All UserId and GroupId values remain the same.

3 Select **Set up per site user tables** to clear the **Shared** check box for all tables in the top grid and the **Update Referenced ID** for all tables in the bottom grid.

4 Click **Process** to drop the views at the non-master sites, create the tables and triggers at the non-master sites, copy information from the master site to the tables at the other sites, and regenerate the replication triggers for all sites. (Dropping the views and creating the tables does not occur in a non-master site that is in the same application database as the master site.)

If you have many sites on this intranet or many users and groups, processing may take a while. The **Processing Step** area displays the system's progress.

During processing, the system validates link setup between the master site and the using sites of an intranet. If it finds a problem, an error message displays and nothing is processed; fix the link and then click Process again.

The **Processed** field indicates which tables have been processed. (Once a row on the form is marked as Processed, subsequent "Process" runs will not reprocess that row.)

The **Status** field indicates that the user tables are not shared.

5 After processing is complete, you must reapply a valid license document on the master site.

Reverting an _All Table

NOTE: If the site where you unshare the table is in the same database as the master site, there is no view to drop, and the table already exists.

Log in to the master site, where the shared tables exist for the intranet. Then follow these steps:

1 On the **Intranet Shared Tables** form, select this site's intranet.

A list of the _all tables that can be shared displays. (Not every _all table is listed; some are not available for sharing.)

2 Tables that are currently shared between all sites on the intranet are marked as **Shared** and **Processed**.

3 Clear the **Shared** field for all the tables that you no longer want to share.

4 Select **Actions > Save**.

5 Click **Process** to delete the views into the selected _all tables from the other sites, copy the selected _all tables from the master site back to all the other sites, and regenerate the replication triggers for the other sites. If you have selected several tables and have many sites on this intranet, processing may take a while. The **Processing Step** area displays the system's progress.

During processing, the system validates link setup between the master site and the using sites of an intranet. If it finds a problem, an error message displays and nothing is processed; fix the link and then click Process again.

The **Processed** field indicates which tables have been processed - for example, if you just unshared the tables, they are now resident in all the site databases in the intranet. (Once a row on the form is marked as Processed, subsequent "Process" runs will not reprocess that row.)

After processing all the tables and sites, the system regenerates the replication triggers at the master site.

Using the Update _All Tables Form

In a system with multiple sites in different databases, use the **Update _All Tables** form to populate _all tables at the local site or at other sites on the same intranet. You can also use this form to truncate the tables or to delete site (or non-site) records. Note that replication rules for the selected tables must exist between the sites in order for _all table population to occur.

NOTE: This form lists only those tables that are shareable. If all of the internal sites (that is, sites on an intranet that is not set to “External”) which are listed on the **Sites** form reside in the same application database, this should be an empty list.

Caution: Use this form with great care. See additional information about using the Update _All Tables form in the *Replication Reference* guide.

If any errors occur during processing of a table, the error is displayed in the **Message** field.

Populating Values for Other Sites in the Local _All Tables

1. Click the Filter-In-Place toolbar button to display all of the available _all tables.
2. Select the tables you want to populate. You can do this using any of these methods:
   - Manually click the **Select** field for specific tables.
   - Select a category from the **Replication Category** list and click **Select By Category** to select only those tables.
   - Click **Select All** to select all categories.
3. Optionally, select Disable Replication.
4. Either select Include All Sites in Intranet, or select a specific site from the **Site** list to affect that site's rows in the local _all tables.
5. Click either **Repopulate Tables** or **Truncate Tables**.

Truncating _All Tables: Example

If the item_mst_all table is corrupt in a system with non-master sites in separate databases, you could use the **Truncate Tables** button this way:

1. Select item_mst_all in Site A and click the Truncate Tables button. This truncates the item_mst_all table in Site A.

2. Select item_mst_all in Site B and click the Truncate Tables button. This truncates the item_mst_all table in Site B.
3. Select item_mst_all in Site A and click the Repopulate Tables button (leave the Disable Replication check box cleared). This adds all of item_mst records where site_ref = ‘Site B’ to the item_mst_all table in both sites A and B (assuming that site B is replicating item_mst_all table to Site A).
4. Select item_mst_all in Site B and click Repopulate (leave the Disable Replication check box cleared). This adds all of site B's item_mst_all records in both sites A and B.

In this example, either site can have bad data, or both sites can have bad data. The item_mst_all tables in both sites are repopulated entirely from scratch.

Note: The Truncate button only truncates if there is an actual table in the current site. This is true in the master site of a shared environment, or in a multi-site, multi-database environment. In an environment that has a shared view, a delete on the current site is performed instead. If you want to perform a delete on another site, use the delete buttons instead of the **Truncate** button.

Deleting Records from the _All Tables

Click **Delete Site Records** to delete the rows (records) for the specified site (from the **Site** field) in the selected _all tables.

Click **Delete Non-Site Records** to delete the rows (records) that *do not match* the specified site (from the **Site** field) in the selected _all tables.

Populating the Local _All Parameter Tables

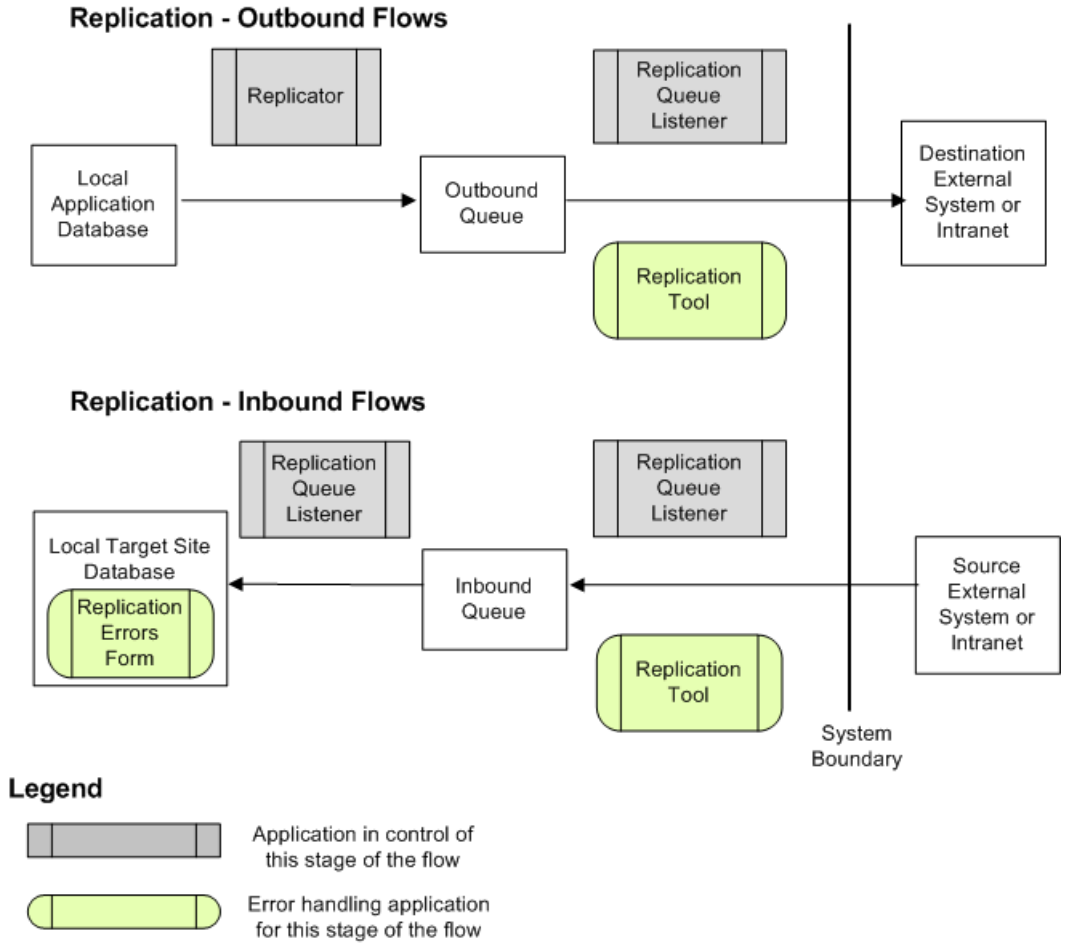
In an initialized database, the parameters tables already have one record defined for the local site. Because the record already exists, it will not be created at other sites through normal replication. Repopulating the tables ensures that a parameter record for the local site exists at other sites where it is needed.

To populate the _all tables in the local site, which is usually done only during multi-site setup:

1. Click the Filter-In-Place toolbar button to display all of the available _all tables.
2. In the **Replication Category** list, select **Initialize _All Parameters** and click **Select By Category** to select only those parameter tables.
3. Make sure nothing else is selected (including Disable Replication or a Site name).
4. Click **Repopulate Tables** to repopulate each selected parameters _all table at the local site and to create or repopulate the local site record at any other sites/entities to which the local site is replicating a category that contains the parameter table.

Handling Replication Data and Communication Errors

If errors occur when sending or receiving XML request documents through this application's replication system, the following processes handle the errors.



Replication Tool (Inbound or Outbound Flow)

If errors happen while replication is processing an inbound or outbound request, you can use the Replication Tool on the utility server to:

- View, correct, and resubmit inbound and outbound XML request documents
- Configure certain types of replication information, and
- View the status of sites linked to this site for replication.

The Replication Tool can be used to view errors in any XML being sent from this application to a different intranet or external system. To start this tool on the utility server, select **Start>Programs>Infor>Tools>Replication Tool**. Online help is available for the tool.

Replication Errors Form (Inbound Flow)

If an inbound XML document makes it into an intranet, is retrieved from the inbound MSMQ by the Replication Listener, but fails when executed against the target site (that is, during application database updates), its errors are displayed in the Replication Errors form on the target site. Generally these documents contain valid login and target site information but are failing for another reason. In the Replication Errors form, you can view and correct the errors, and then resubmit the request.

If an inbound XML document fails before that point, you can view and correct it through the Replication Service Management Tool described above.

Processing Requests in the External Application

After this application posts an XML request to the designated URL for an external application, this application no longer controls what happens to the request. It is up to the external application to process the request. However, you can set up the external application so that it sends status messages to the External System Transaction Log. For example, you might set up the application to return status information about whether the transaction was successfully processed.

BODs and Replication Documents

Replication Documents Overviews & Procedures

This module contains help topics that describe the Replication Document overviews and procedures.

Creating and Maintaining Business Object Documents (BODs)

Use the following process to build or change a Business Object Document (BOD) XML via metadata defined in system forms. BOD elements are often mapped to objects in system tables or collections, although certain elements may be specified as literals or calculated values that are not tied to any table or collection.

In the Replication Documents form, you specify the general structure and conventions used by the BOD. Specify when the BOD is used through either the Replication Document Outbound Cross References form or the Replication Document Inbound Cross References. Then use the Replication Document Elements and Replication Document Attributes forms to define the elements (BOD tags) and the attributes of those elements, and map the elements and attributes as required.

BOD Structure and Usage

1. On the **Replication Documents** form, specify the document name, used only for internal reference. The actual BOD XML document name is specified by the BOD Noun and BOD Verb fields in the Primary Tab.
2. Enter information about the IDO used to build the BOD. If no existing IDO collection fits the purpose of the BOD, you can define a new IDO collection or custom load method specifically for the BOD.
3. Click either the **Inbound Cross-Refs** or **Outbound Cross-Refs** button to go to one of the following forms, where you can specify how the BOD is used:
   - For outbound BODs (those generated by this application), use the **Replication Document Outbound Cross References** form to enter information about how this BOD is triggered.

     Examples

     NOTE: Do not use this option if this application is used only to facilitate integration between two other applications.

- For inbound BODs (those generated by some external system that will update some data in this application), use the **Replication Document Inbound Cross References** form to enter information about which replication document to use with a particular BOD Noun and Verb.

BOD Header

NOTE: This section applies only to BODs intended for applications that use the standard Infor BOD format. Generic BOD XMLs intended for other applications do not require these BOD header elements.

1. On the **Replication Documents** form, click the **Add BOD Headers Element** button to automatically generate the header elements required for any BOD.
2. The **Replication Document Elements** form displays with the appropriate header elements added.

   The *BOD_Name*/ApplicationArea/BODID element contains the NID in this format:

   ```
   NIDID(DerBODID)?BODNOUN()&verb=BODVERB()
   ```

   In most cases, you do not need to edit this element. However, if you want the NID's location value populated, change the NIDID macro to LNIDID. (Click here for more information about these macros.)

   When you create the associated IDO or Custom Load Method used to populate the BOD, you must include a property named DerBODID that contains the key value for the BOD. For example, for the PurchaseOrder BOD, you must define a DerBODID property that contains the purchase order number.

   Save the records and close the form.
3. The **Attributes** form displays with the appropriate attributes automatically defined for the header elements. Save the records and close this form.
4. If this BOD is used to publish deleted data, change the ValueExpresson of the ActionCode attribute on the ActionExpression element of the header from "Replace" to "Change".

BOD Body

1. On the **Replication Documents** form, click the **Elements** button to open the **Replication Document Elements** form again. This time you will be adding elements to the body of the BOD, to produce all the necessary tags for the BOD.
2. For each element (tag), set up the following:
   - For the **BOD Tag Name**, enter the actual tag included in the XML.

- Specify whether the value of the element is a literal value, a dynamic subcollection, or a property (that is, mapped to a data element in an IDO collection). If you are adding a dynamic subcollection, see Adding a Collection of Elements from a SQL Table or IDO to a BOD for more information.
- If the value is a literal, enter the **Value Expression**. If the value is a dynamic subcollection, enter the **Dynamic Subcollection IDO Name**. If the value is a property, enter the **Property Name**. The property name you enter is a property on the IDO listed on the parent BOD form, or a property on a subcollection IDO linked to that IDO.
- If the value is an explicit "To" logical ID, select Is To Logical ID.
- On the Subcollections tab, specify information about any custom load methods, filters, overrides, or Link By information for subcollections.

3 If any element includes an attribute - that is, the BOD tag looks like this:

```
<element attribute=xxx>
```

4 then click the **Attributes** button to display the **Replication Document Attributes** form, filtered for the current element. (Notice that the element's sequence number is displayed for reference.)

5 Specify the following for the attribute:

- for the **Attribute Name**, enter the actual attribute included in the XML. The attribute name is case-sensitive.
- Specify whether the value of the attribute is a literal value or a property (that is, mapped to a data element in an IDO collection).
- If the value is a literal, enter the **Value Expression**. If the value is a property, enter the **Property Name**. The property name you enter is a property on the IDO listed on the parent BOD form, or a property on a subcollection IDO linked to that IDO.

Tips for Constructing the Body

- An XML document inbound to this application from an application that does not use the Infor BOD format must include the following XML element:

```
<DocumentData TargetSite="target_site" SourceSite="source_site" Verb="verb"
Noun="noun">
```

- When building BODs for applications that do use the Infor BOD format, be aware of the following:
- For the *main ID element* in the Noun section of the BOD (this is often, but not always, called **DocumentID**):

  - Specify the **BOD Tag Name**.
  - Set **Value Type** to **Literal**.
  - Set **Property Name** to **DerBODID**.
- For *reference elements*, the ID should be populated with the key value (DerBODID) used by the BOD being referenced.
- If the BOD is used to publish deleted data, only two elementsin the Noun section of the body need to be mapped: the main ID element and the status element.
- All values published in an ID element or reference ID element must have leading spaces trimmed. If an ID element is populated with concatenated columns of data each column of data must have leading spaces trimmed.

  The following example shows concenation and trim logic:

  ltrim(VendVch.vend_num) + '~' + ltrim(cast(VendVch.voucher as nvarchar(7)))

  If vend_num = 200 and voucher = 5000 then the value published would look like this:

  200~5000
- If a parent tag contains no data and has no attributes, you do not need to define an element record for it in the metadata. If a child tag element is defined in the Elements form, the system automatically generates the parent tag in the XML. For example, if you define metadata for

  ProcessCustomerPartyMaster/DataArea/CustomerPartyMaster/Location/Name

  but you have not defined metadata for the parent

  ProcessCustomerPartyMaster/DataArea/CustomerPartyMaster/Location

  the system includes any missing parent tags in the XML, like this:

```
<DataArea>
  <CustomerPartyMaster>
     <Location>
       <Name>MyCo</Name>
     </Location>
  </CustomerPartyMaster>
</DataArea>
```

- When an element is populated with multiple columns of data concatenated together, the approved concatenation character is ~ (tilde). For example, if concatenating cust_num and cust_seq together to be published together as one value in an element, specify it as follows:

  cust_num~cust_seq

If cust_num = C000001 and cust_seq = 5 then the value published looks like this:

C000001~5

- Use the Documentation tabs and fields on the various forms to enter implementation information for the BOD and its elements and attributes. This information can be used to create data mapping/implementation reports.
- You can create a Custom Load Method whose source is a view that contains all the information you need in the BOD. The CLM returns the result set of the view. This may produce speed improvements over using an IDO.

  NOTE: If this application is used only to facilitate integration between two other applications, then the CLM in this application runs a corresponding CLM (via a remote method call) in the actual application.

Next Step

After you define the replication document mappings for a BOD, you must set up replication for it. See Replicating Data as BODs to Other Infor BOD-Enabled Applications. Or, if you are defining generic (non-Infor) BOD XML documents, see Replicating Data to External Systems That Do Not Use Infor BOD Formatted XMLs.

If you update a replication document for a BOD that has been replicated previously, you should restart the ReplQListener service to clear the cache of old metadata for the BOD.

Generating Replication Document Scripts

Use the **Repl Doc Script Generator** form if you want to generate a replication document script and then, optionally, check the script into a source code control system. This allows you to include your custom replication document information in system upgrades.

Examples: Function- and Table-Based BOD Generation

This topic explains how to set up BOD generation triggers. This is a task that is part of these other larger procedures:

- Replicating Data as BODs to Other Infor BOD-Enabled Applications - Replication Categories step
- Creating and Maintaining BODs - Replication Document Outbound Cross References step

There are two ways to trigger the generation of a BOD using replication categories and replication document outbound cross-references:

- **Table** - The simplest way is to specify the table or view in the Replication Document Outbound Cross-References form for the BOD, and then add the table name to a Replication Category. Any change made to a record in that table triggers generation of a BOD.

If you do not want a BOD to be generated every time anything in that table is changed, you can restrict BOD generation using fields on the **Replication Categories** form:

- The Filter attribute: you can define a filter that determines if a change to the table must trigger BOD generation. Only records that match the filter are included.
- Checkboxes that restrict BOD generation based on the type of change to the table: Skip Insert, Skip Update and Skip Delete.
- **Function** - Use this technique when a simple table-update based trigger is inadequate. For example, suppose you want to trigger a BOD when a new order is created. Since an order is hierarchical data and may have any number of lines and releases, it would not be possible to identify a single table that would trigger the BOD generation; there is no way to know when the last line or release has been inserted. Or you may need to trigger BOD generation when some other application event occurs that is completely unrelated to a table update. The solution is to programmatically trigger the BOD update using a "Function" trigger.

  In the **Replication Categories** form and the **Replication Document Outbound Cross-References** form, you specify a "Function" name. This name does not necessarily correspond to an actual stored procedure, and likely it will not. It is really just a parameter that is passed into a replication stored procedure that ultimately triggers the BOD generation. (For more information about how this works, see Behind the Scenes: How the System Generates a BOD.)

  You could also call this stored procedure from a trigger, another SP, or an IDO extension class as well (and probably other contexts if necessary).

If you plan to generate different XML documents for different target sites, you can include the appropriate target site's Intranet Name on the replication document cross-reference record.

Function Trigger Setup Example

Whenever the "function" TriggerSalesOrderSyncSp is invoked by some system action, you want your BOD to be generated.

In the **Replication Document Outbound Cross-References** form, you specify this trigger information:

- Applies to IDO: **SP!**
- Applies to IDO Action: **Invoke**
- Applies to Method: **TriggerSalesOrderSyncSp**

In the **Replication Categories** form, you specify this trigger information:

- Table or Function: TriggerSalesOrderSyncSp
- Object Type: SP
- Filter: Leave this field blank.

Define the rest of the BOD and replication setup as specified in the procedure topics listed above.

Table Trigger Setup Example

You want your BOD to be triggered by a change to the employee view on the employee_mst table, according to the filters and checkboxes you specify in the **Replication Categories** form.

In the **Replication Document Outbound Cross-References** form, you specify this trigger information:

- Applies to IDO: **TABLE!employee**
- Applies to IDO Action: **UpdateCollection**
- Applies to Method: **N/A**

In the **Replication Categories** form, you specify this trigger information:

- Table or Function: **employee**
- Object Type: **Table**
- Filter: Specify any appropriate filter.

Set the rest of the columns as needed.

NOTE: For table triggers, you must select the **Skip Delete** check box. Otherwise, a BOD will be created with null values on deletion.

Define the rest of the BOD and replication setup as specified in the procedure topics listed above. If you use non-primary key fields from the UpdateCollection (column names) as properties, then you must set the **Update All Columns** checkbox in the **Replication Rules** form. When you add a replication rule for a table, you must regenerate the replication triggers.

Adding Elements from a SQL Table or IDO to a BOD

You might want to modify an existing BOD to add elements based on columns in a table or an IDO. To add the elements through the **Replication Document Elements** form:

1. Select the BOD on the **Replication Documents** form.
2. Open the **Replication Document Elements** form linked to the BOD.
3. Create a new element record with these values in the Primary tab:
   - BOD Tag Name: Leave this field blank or enter a tag name for the element group. This tag name will be an empty element, but it can be used to designate the tag start and ending point in the resulting BOD if **Include Empty Element** is selected.
   - Value Type: Set to **Dynamic Subcollection**.
   - Dyamic IDO Subcollection Name: Specify either a table (TABLE! plus the table or viewname, for example **TABLE!itemcust**) or specify an existing IDO (for example, **SLItemcusts**).
4. In the Subcollection tab, set the Link Value to the column in the table from which the elements are taken, for example **itemcust.item=item**, which then links the table subcollection to the primary collection where **item=item**, or **SLItemcusts.item=item** for the IDO. Additional filter statements can be added to the Filter field.

Note: A collection name must be unique within the BOD. You cannot access TABLE!itemcust filtered on one record in the collection and then access it again on another.

When you generate the BOD, it includes an element for each value in the table/view or IDO that matches the Link By and Filter statements. In our example, there would be an <itemcust> element generated for each item value in the BOD.

Viewing Received and Sent Business Object Documents (BODs)

Business Object Documents (BODs) generated by the application are placed in a system outbox, where they are retrieved and sent them to other BOD-enabled applications.

BODs sent by some external applications to this application are placed by replication into a system inbox.

You can view the following information about existing BODs in the Replication Document Inbox and Replication Document Outbox forms:

- Whether the document has been picked up for processing
- When the document was created
- What system created it
- What type of BOD it is (noun and verb)
- The content of the XML in a read-only browser pane

From either of these forms, you can select a specific document and click the **Export XML** button to save the XML to a filename you specify.

NOTE: If you need to view the documents in the outbox, make sure you configure the BOD Loader so that it does not delete the documents from the outbox after processing. See the BOD Loader documentation for more information.

Replicating Data as BODs to Other Infor BOD-Enabled Applications

This application can use BODs in order to replicate data to and from other BOD-enabled applications.

Before You Start

- Make sure you have all the necessary software installed and configured. For more information, refer to the appropriate installation and integration guides.
- If you are using custom business object documents (BODs), make sure they are defined using the Replication Document forms.

- Be aware that this process does not currently apply to Applications that are not Infor BOD-enabled - See Replicating Data to External Systems That Do Not Use Infor BOD Formatted XMLs.

Set Up This Application to Work with BODs

1 If multiple sites on an intranet will send or receive BODs, specify the "bootstrap" configurations on the Replication tab of the Service Configuration Manager utility, on the utility server:

   - Specify a **Replcator Configuration** that is the collection site for all **outbound** messages from sites on the intranet. This site's **Replication Document Outbox** will collect BODs for all sites on the intranet.
   - Specify an **Inbound Bus Configuration** that is the collection site for all **inbound** messages from sites on the intranet. This site's **Replication Document Outbox** will collect BODs for all sites on the intranet.

   These two configurations can point to the same site database. In many cases, the inbound and outbound sites will be the same site.

2 On the **Intranets** form, create a new intranet for Infor ION (bus). Specify these values:

   - **Intranet:** Specify the name to identify the intranet, for example: **InforBUS**
   - **Description:** Specify an appropriate description, for example: **Infor ION connection**
   - **External:** Select this check box.
   - **Transport:** Select **ESB**.
   - **Tenant ID:** To use the same tenant ID for all sites on this intranet, specify it here. You can also leave it blank here and specify the tenant ID on the local site records, or leave that field blank to use the default tenant ID value of **infor**. The tenant ID can be a maximum of 22 characters. For more information about how the tenant ID is used with ION, see the *ION Connect Administration Guide*.

3 On the **Sites** form, create a logical "site" for Infor ION:

   - **Site:** Specify a name to identify this site as your ION site, for example: InforBUS.
   - **Site Name and Description:** Specify an appropriate site name and description.
   - **Intranet Name:** Select the intranet you created in Step 1.
   - **From Site:** On the Site User Map tab, specify the local site, for example, **oh**.
   - **User Name:** Specify the user ID used to send replication documents to the Mongoose outbox for ION to retrieve. This user must already be set up on the **Users** form at the local site. We recommend that you specify the repl_user here, if it is defined. Otherwise, specify a user with Full User editing permissions.

4 On the **Sites** form, configure the local site record to generate and/or receive business object documents (BODs):

   - Specify the **Message Bus Logical ID** for the site. This identifies the Mongoose site to ION. Make sure it is set up as specified in Formatting Infor Logical IDs.

     Note: The logical ID specified here must match the application type and instance defined in ION Connect for this site.

- (Optionally) Specify a tenant ID for the local site. If no tenant ID is specified here or in the **Intranets** form, the tenant ID defaults to **infor**.

5 On the **Replication Categories** form, the **ESB** category is used for replicating BODs.

If you are adding a *new* BOD, you must add to the ESB category the table or method being used to generate the BOD.

Examples

NOTE: For BODs created with TABLE!, you must select the **Skip Delete** check box. Otherwise, a BOD will be created with null values on deletion.

6 On the **Replication Rules** form, create a new rule:

a **Source Site:** Specify the local site that you configured in Step 4.

b **Target Site:** Specify the ION site you created in Step 3.

c **Category:** Specify **ESB**.

d **Interval Type:** Specify any option *except* **Transactional**.

7 On the Replication Management form, click Regenerate Replication Triggers.

The next time a user action triggers BOD generation, the BOD XML will be created automatically and placed in the **Replication Document Outbox**.

Generating Specific BODs through a Utility

In some cases (for example, when initializing an integration), you may need to manually generate certain BODs. You can use the Replication Document Manual Request Utility to do this.

Preventing a BOD From Replicating

If you have a BOD that is still under development, you can include it in the ESB replication category but select Skip Method or Skip Table, as appropriate, to prevent it from replicating.

Behind the Scenes: How the System Generates a BOD

These are the steps the system goes through in generating a BOD.

1 This application uses its replication engine to generate every BOD. Two separate processes can trigger BOD generation:

- **Table generation:** A change is made to a table that triggers an UpdateCollection request.
- **Function generation:** A business event occurs in the system that triggers a remote method call.

Each of these is considered a *BOD generation integration point*. The business events that generate a BOD are listed in the **Replication Document Outbound Cross References** form.

Example 1 - Table Generation: When you add a table name to a replication category, then any change made to a record in that table triggers a BOD generation. If that is too general - that is, you do not want a BOD to be generated every time something touches anything in that table - you can restrict BOD generation by using the Filter attribute to define a filter that determines if a change to the table will trigger BOD generation. Only records that match the filter are included. There are also three check boxes that you can use to restrict BOD generation based on the type of change to the table – Skip Insert, Skip Update and Skip Delete. All of this is done through the **Replication Categories** form.

Example 2 - Function Generation: Use this technique when a simple table-update based trigger is inadequate. For example, suppose you want to trigger a BOD when a new purchase order is created. Since a purchase order is hierarchical data and can have any number of lines and releases, it is not possible to identify a single table that triggers the BOD generation - there is no way to know when the last line or release has been inserted.

As another example, if you want to trigger BOD generation when some application event occurs that is completely unrelated to a table update, you can programmatically trigger the BOD update using a "Function" trigger. The "Function" name specified in the **Replication Categories** form does not necessarily correspond to an actual stored procedure, and likely it will not. It is really just a parameter that is passed into one of the replication stored procedures that ultimately triggers the BOD generation. The stored procedure is named RemoteMethodForReplicationTargetSp. You could also call this stored procedure from a trigger, another SP, or an IDO extension class.

To illustrate the purchase order trigger mentioned above: A user runs the Purchase Order Report to create a new purchase order. As a part of the logic executed, a remote method call is made to the stored procedure **TriggerPurchaseOrderSyncSp**, which triggers BOD generation replication logic.

Stored procedures called via a remote method call are not actual stored procedures that exist in the database. The call may or may not pass parameters. If the call does pass parameters, the Replication Document forms will refer to these by sequence; that is, **P1** refers to the first parameter, **P2** to the second parameter, and so on. The following code snippet shows the two parameters passed by a remote method call to TriggerPurchaseOrderSyncSp:

```
EXEC @Severity  = dbo.RemoteMethodForReplicationTargetsSp

  @IdoName        = 'SL.ESBSLPos'
, @MethodName = 'TriggerPurchaseOrderSyncSp'
, @Infobar          = @Infobar OUTPUT
, @Parm1Value   = @PoPoNum
, @Parm2Value   = @Infobar
```

2 In the Replication Categories form, each category contains all of the tables and functions required for replication of a particular area of the system. A category called **ESB** contains all the defined remote method calls, or Object Names referred to by Replication Documents, for

integration points to other applications that require XMLs in the Infor BOD format. For example, TriggerPurchaseOrderSyncSp is listed in the ESB category.

For generic XML documents (that is, documents intended for applications requiring XMLs not in the Infor BOD format), you would need to create a separate replication category that contains the appropriate remote method calls or object names.

3 If a replication rule exists between a local site and an Infor ESB site where the category is set to **ESB**, then whenever a business event or table change with an integration point occurs at the local site, the system adds an entry in the Replication tables.

For generic XML documents, you would create a replication rule between a local site and the site defined for the application. That site must be set up on an intranet that is defined to use HTTP transport protocol.

4 The Replicator picks up rows from the Replication Tables and places them in MSMQ.

5 ReplQListener picks up the rows from MSMQ and determines that these are rows bound for other BOD-enabled applications. It then generates a BOD for the business event or table change. As mentioned above, parameters may be passed by the event.

In our Example 2 above, a user runs the Purchase Order Report, which calls TriggerPurchaseOrderSyncSp. Since a rule exists for a category where this Sp is defined, the system builds a replication document. It finds a replication cross reference for **PurchaseOrder.Sync** where the Applies To method is TriggerPurchaseOrderSyncSp, so it retrieves the **PurchaseOrder** replication document template and builds the BOD XML document, using the noun and verb from the Replication Document Outbound Cross Reference form, plus element and attribute information defined in the Replication Documents, Replication Document Elements, and Replication Document Attributes forms.

Also in this example, the purchase order number is passed as the first parameter (P1). The replication document uses this information in a filter (PoNum=FP(P1)) so that the generated BOD contains data for only the specified purchase order.

6 ReplQueueListener places the generated BOD XML in the Replication Document Outbox.

Generic XML documents, instead of going to the Outbox, are sent to the intranet URL that is defined for non-transactional replication.

Scenarios, Guidelines, and an Example using the Exclusion Property Name in BOD Definitions

Scenarios

There is no way for the BOD generation logic to automatically determine when an element must be published that is listed as mandatory in the Infor defined BOD spreadsheets. Also, depending on how this new exclusion functionality is used, the resulting BOD may or may not bring about the desired result. Thus, it is left up to the BOD developer to make sure your use of the Exclusion

Property Name field is consistent with the Infor BOD requirements. Evaluate each case to ensure that you achieve the result you expect. The following scenarios can help you determine how to exclude elements.

Use these definitions when reading the scenarios:

Parent = parent element, defined in the Replication Document Elements form.

Child = child element of parent, defined in the Replication Document Elements form.

Mapped = element has a property in the Replication Document Elements form Property Name field.

Not mapped = element does not have a property in the Replication Document Elements form Property Name field.

propA = a property name, which may or may not have a value, stored in the Replication Document Elements form Property Name field.

propB = a property name, which may or may not have a value, stored in the Replication Document Elements form Property Name field.

EPN = Exclusion Property Name field on the Replication Document Elements form.

Scenario 1: Parent, not mapped; Child, mapped to propA

- If the parent element must have attributes then both the parent and child EPN must be set to propA.
- If the parent element must have a child element then both the parent and child EPN must be set to propA.
- If the child element must have a value then the child EPN must be set to propA.
- If the parent must have attributes and the parent must have a child and the child must have a value then both the parent and child EPN must be set to propA.
- If the parent can be included without attributes and the parent can be included without a child element and the child element can be included without a value, then both the parent and child EPN can be set to any property name.

Scenario 2: Parent, mapped to propA; Child, mapped to propB

- If the parent must have a value, the EPN of the parent must be set to propA.
- If the parent must have attributes, both the parent and child EPN must be set to the same property.
- If the parent must have a child, both the parent and child EPN must be set to the same property.
- If the child must have a value, the child EPN must be set to propB.
- If the parent must have a value, attributes and a child, both the parent and child EPN must be set to propA.

- If the parent can be included without a value, without attributes and without a child, and the child can be included without a value, the EPN for both the parent and child can be set to any property.
- Neither element can be conditionally excluded if the parent must have a value, have attributes, have a child and the child must have a value.

Guidelines

Follow these guidelines for using the Exclusion Property Name component:

- There is probably never a good reason to define an exclusion property for any element in the ApplicationArea of the BOD.
- Do not define an exclusion property for an element that is used to declare a subcollection.
- You can use a derived property as an exclusion property in the absence of another property that can be relied upon to control exclusion. The derived property may, or may not, be used elsewhere in the BOD metadata mapping.

Example

Parent, not mapped, with the attribute type='123', EPN=prop1

Child, mapped to prop1, EPN=prop1

Given that prop1 had the value of ABC, then the resulting BOD XML would look like the following:

```
<parent type='123'>
   <child>ABC</child>
</parent>
```

Given the same definition except that prop1 was empty, then neither the parent or child elements would be published.

Deleting Business Object Documents (BODs)

When the BOD Loader retrieves BODs from the Replication Document Outbox, it also removes them from the outbox, unless it has been configured to preserve them. Usually you want the BODs to be deleted automatically, unless you are troubleshooting.

Generating a BOD as a Background Task

Generating a BOD as a Background Task

Introduction

In this application, a BOD is generated automatically by the appropriate *application events* - for example, a change in status or receipt of an item - as defined in the Replication Document forms. In certain cases, a BOD must also be generated by a *scheduled event* rather than an application event.

For example, ProductionOrder BOD generation is triggered by a change of status, but is not triggered by changes in the production schedule. Generating a ProductionOrder BOD each time a date change event occurred would generate so many BODs during a single day that it could overwhelm the system. Thus, using an application event to trigger ProductionOrder BODs on schedule changes is not a good choice. Instead, use a scheduled event that runs as a background task. The generated BOD will include all schedule changes since the last time the BOD generation was run.

Setup for Generating a BOD as a Background Task

1. In the Replication Document Outbound Cross References form, set up a definition for the BOD that requires a scheduled event.
   a. To quickly set up the BOD definition, copy a current definition for the same BOD to the new definition, and assign a new method through the Applies To Method field.
   b. Select Eligible for Background Manual Request so the BOD can be generated as a scheduled background task.
   c. Apply filter conditions as described below.

   NOTE: Filter conditions are required to restrict the number of BODs that are created and to define the time at which they are created, in order not to overwhelm the system during business hours. The ProductionOrder BOD is particularly vulnerable to causing a system overload because of its size and the number of times it can be generated. For examples of filters that are appropriate for production orders, see Example: Setting up a Production Order BOD Generation to Run in the Background.

2. In the Replication Categories form, create a new category that contains the method you created in Step 1, or use the existing **ESB** category.
3. In the Replication Document Manual Request Utility, create a background task that will generate the selected BODs:
   a. Select the appropriate BOD Noun, Verb, and Method.
   b. Select **Update** in the Initial Load/Update field.
   c. From the Activities menu, select **Background**.

NOTE: For the ProductionOrder BOD, we recommend that you run the Planning or Scheduling process before running the background task.

The background task selects the outbound cross reference definition and generates the BOD at the scheduled time. Settings and filters used in the outbound cross reference definition are not saved within the background task, so that you can adjust them and use them for the next scheduled run.

Filter Conditions

Dependent applications such as Infor EAM require different information to be supplied at different times, based on what the application needs to perform or monitor. Use filters to supply only the needed information. When you create a filter, consider the application-generated events that trigger a BOD. For example, a ProductionOrder BOD is automatically generated when a job is created, when a status changes, or when a quantity changes while the job is in "released" status.

A dependent application may not require a production forecast over the next year. External application-dependent conditions such as this can significantly affect the filtering. If the BOD supports several external dependent applications, you can create several different outbound cross reference definitions with application-specific filters. For example, to support an application reporting on work in process, you could set up the ProductionOrder BOD with a filter to show released jobs and a schedule to generate the BOD once a month.

Filters are based on the Data Key Source that supplies the Key Column List. For example, the key source for the ProductionOrder BOD is the ESBProductionOrderView. You can change the key source to a view or to a table that supplies the required Key Column List elements. This allows you to create a complex relation and to filter out other elements based on the view or table, and not be restricted by the current view or table.

If filters are based on conditional data that can change over time, and the data changes outside of that filter range, the information will never be updated when that filter changes. This is particular to dates. To address this condition, the published_start_date field was added to the job_sch table to indicate the last start date used in the ProductionOrder BOD. In the ESBProductionOrderView the field is LastStartDate.

Filters are cumulative in the form of Line 1 AND Line 2.

Examples: Setting up a ProductionOrder BOD Generation to Run in the Background

The **Replication Document Outbound Cross References** form includes three provided definitions for generating ProductionOrder BODs from the application. The definitions have different methods and filters. The methods are:

- TriggerProductionOrderSyncSp - include all production orders
- TriggerProductionOrderBGSyncSp - include all production orders just scheduled
- TriggerProductionOrderBGSchSyncSp - include all production orders that have changed.

The definitions with "BG" in the method name can be run as background tasks. These sample definitions do not include filtering criteria for the date range; you can add date filters after the first time the background task is run, when a date appears in the Last Manual Publish Date.

Example 1

Using the definition containing the method TriggerProductionOrderBGSyncSp and the filters below, all newly created jobs are updated after planning or schedule run. The filter obtains the ProductionOrder after the first schedule update. The job was entered with dates. The planner runs for the first time and updates start and end dates.

Use this combination of filters to limit the Production Order records:

- Filter for firm jobs only; this picks up schedule changes:

  Status = 'Firm'

  You could also filter for released jobs only to pick up material and labor changes for actual reporting.

- Filter for any records that have changed since the last time the background task was run:

  RecordDate > dbo.GetReplDocLastManualPublishDate('SP!', 'Invoke', 'TriggerProductionOrderBGSyncSp')

- Filter for jobs that have been created (jobs in firm status) that have not previously been scheduled:

  CreateDate > dbo.GetReplDocLastManualPublishDate('SP!', 'Invoke', 'TriggerProductionOrderBGSyncSp')

- You can add a filter for certain items:

  Item BETWEEN 'item number' AND 'item number' AND NOT Item = 'item number'

- You can add a filter for certain product codes:

  Product_code IN ('Product_code1','Product_code2')

Example 2

The definition containing the TriggerProductionOrderBGSchSyncSp method can use the filters below, which are specific to a scheduled change within a time fence determined by the user (a planning window of one month in the past and one month in the future).

- Filter for firm jobs only; this picks up schedule changes:

  Status = 'Firm'

- Filter for any records changed since the last time the background task was run:

  RecordDate > dbo.GetReplDocLastManualPublishDate('SP!', 'Invoke', 'TriggerProductionOrderBGSchSyncSp')

- You also filter for jobs that have been changed since the last ProductionOrder scheduled start date or since the current scheduled start date within the time fence:

MONTH(StartDate) BETWEEN MONTH(GETDATE() - 1) AND (MONTH(GETDATE())+ 1) OR MONTH(LastStartDate) BETWEEN MONTH(GETDATE() - 1) AND (MONTH(GETDATE()) + 1)

- You can add a filter for certain items:

  Item BETWEEN 'item number' AND 'item number' AND NOT Item = 'item number'

- You can also add a filter for certain product codes:

  ProductCode IN ('Product_code1','Product_code2')

- You can add a filter to not create BODs that were created in the first scheduled run:

  CreateDate < dbo.GetReplDocLastManualPublishDate('SP!', 'Invoke', 'TriggerProductionOrderBGSyncSp')

Reverting to the Default Version of a Form

If you have customized a form and you want to revert to the original (default) version, consider these questions:

- Have you verified that the form has, in fact, been customized?

  To see whether a form has been customized, select **Help > About This Form**. If the form version is not "Vendor Default" then the form has been customized.

- Who are you?

  If you are not a vendor developer or a site developer, you can revert only those changes that you personally have made to the form.

- What do you want to revert to?

  If you have site developer, full user, or basic user editing permissions, you can cancel customizations to the form and any associated global form objects and restore another version according to this hierarchy:

  - A user version reverts to a group version (if one exists).
  - A group version reverts to a site version (if one exists).
  - A site version reverts to the default vendor version.

- Do you want to save a copy of this version?

  If you have site developer editing permissions, you can copy the customized form to a new form name before you revert the form defintion. Then, after reverting the existing form, you can copy any customizations that you want to keep, from the copy to the new standard form.

  To make a copy of a form, **select Form > Definition > Copy** and then enter the Source and Target form names. For example, you could enter CustomerOrderLines for the Source form and Site_CustomerOrderLines for the Target form.

Reverting changes to a form made in Runtime Mode

To revert run-time changes made to a form:

1. Open the form in Runtime Mode.
2. Click the "Revert runtime changes" button on the main toolbar.
3. When prompted for confirmation, click **OK**.

   WinStudio closes the form, reverts to the vendor default version, and reopens the form.
4. Verify that the run-time changes have been reverted.

Reverting a form to the vendor default form definition

NOTE: This procedure applies only to customizations and modifications made at any level in Design Mode.

To revert a form to the vendor default form definition:

1. Open the form in Design Mode.
2. Select **Edit > Revert Form Definition**.
3. When prompted for confirmation, click **OK**.
4. Close the form and select **Form > Definition > Unload All Global Form Objects**.
5. Reopen the form and verify that it has reverted to the vendor default form definition.

About Maximum Concurrent Tasks

To avoid overburdening TaskMan, you can limit the total number of tasks being run at the same time. This is accomplished through the use of three fields:

- Maximum Concurrent Tasks and Maximum Concurrent Report Tasks on the **Intranets** form
- Max Concurrent on the **Background Task Definitions** form

NOTE: If a task is submitted, but the maximum number of tasks is currently running, the task goes into a queue, and is processed as soon as a place becomes available.

Maximum Concurrent Tasks

This field is used to define the maximum number of all tasks, including report tasks, that can run at a single time on the current configuration. If a task is submitted, TaskMan processes it only if the total number of tasks currently running is less than this number.

Multiple Configurations Monitored by TaskMan

If TaskMan is monitoring multiple configurations, then each configuration's number of maximum tasks combine to define the maximum number of tasks that this specific TaskMan service can handle.

EXAMPLE:
TaskMan is monitoring two configurations: **C1** and **C2**
Maximum Concurrent Tasks for C1: **10**
Maximum Concurrent Tasks for C2: **20**
Maximum Concurrent Tasks for this instance of TaskMan: 10 + 20 = **30**

- If 10 tasks are running on C1, then C1 can no longer run any tasks, because its limit has been met, even though TaskMan can handle 20 more tasks.
- If 20 tasks are running on C2, then C2 can not launch any more tasks, but C1 could launch up to 10.

Maximum Concurrent Report Tasks

This field is used to define the maximum number of report preview or print report tasks that can run at a single time on the current configuration. If a report or report preview is submitted, TaskMan

process it only if the total number of report previews or print jobs currently running is less than this number.

NOTE: TaskMan tracks report preview and print report tasks separately. This means, for example, that if you have this field set to **20**, then you can have up to 20 report preview and 20 print report tasks processing simultaneously.

Because reports tasks are a sub-set of all tasks, Maximum Concurrent Report Tasks should be less than, or equal to, **Maximum Concurrent Tasks**. If you allow more reports than overall tasks, TaskMan limits the reports to the maximum overall tasks.

EXAMPLE:
Maximum Concurrent Tasks: **10**
Maximum Concurrent Report Tasks: **20**

- If you submit 20 reports, only 10 run, because that is the maximum number of overall tasks.
- If you submit 20 reports while 5 tasks are already running, only 5 reports are processed immediately.

Multiple Configurations Monitored by TaskMan

If TaskMan is monitoring multiple configurations, then each configuration's number of currently running report tasks is calculated by adding the currently running report tasks from each configuration that TaskMan is monitoring.

EXAMPLE:
TaskMan is monitoring two configurations: **C1** and **C2**
Maximum Concurrent ReportTasks for C1: **10**
Maximum Concurrent Report Tasks for C2: **20**
Number of reports currently running on C1: **5**
Number of reports currently running on C2: **3**
Number of reports tasks available to run on C1: 10 - (5 + 3) = **2**
Number of reports tasks available to run on C2: 20 - (5 + 3) = **12**

Max Concurrent

This field is used to limit the number of times a single, specific task can be run at the same time.

EXAMPLE:
Maximum Concurrent Tasks: **10**
Maximum Concurrent Report Tasks: **5**
Max Concurrent for the ABCAnalysisRpt task: **1**

- If no tasks are currently running, and you submit the ABCAnalysisRpt report task twice, only one will be processed immediately.
- If five report tasks are currently running, and you submit the ABCAnalysisRpt report task, it will not run until one of the five already running has finished.

- If eight general tasks, and two report tasks, are running, and you submit the ABCAnalysisRpt report task, it will not run immediately, even though only two of the available five report tasks is running, because the total number all tasks has been met.

Scheduling Background Tasks (Developer/Administrator)

The application runs reports as background tasks that are placed on a "queue" to be run in order. You can also set up other tasks to run in the background. The TaskMan Windows service polls the list of Active Background Tasks and executes any new task that is posted to the queue with a status of READY.

SQL Server Agent must be running on the database server in order to perform background processing.

For more information about TaskMan, see the Infor Mongoose Administration Guide.

Create a Background Task

1. Create a new record on the Background Task Definitions form.
2. Specify a task name (for example, RunCustomerOrderReport) and description.
3. Specify an executable:
   - For a report, specify the name of the report (for example, CustomerOrder) in the Executable Name field and the type of executable (RPT) in the Executable Type field.
   - For a stored procedure, specify the procedure name in the Executable Name field and select SP in the Executable Type field.
   - For an executable program, specify the program name and path (for example, c:\Infor\myprog) in the Executable Name field and select EXE in the Executable Type field.
   - For an IDO method, specify the method name (in the format IDO.Method - for example, MyProgID.SLSites.MyTestMethod) in the Executable Name field and select IDOMTH in the Executable Type field.
4. Click the buttons on the form to specify any report options or excluded tasks.
5. Save the record.

Your new background task can be called from any form as an event handler.

Check to See if the Task is Running

When TaskMan starts a task, it updates the Started field in the Status Information group box. If the Completed field is blank, the task is still running.

Also, you can see which tasks are currently running by opening the Active Background Tasks form.

View Details About the Completed Task

After TaskMan has completed the task, even if it fails, you can view details about its execution on the Background Task History form. Details include:

- Task description
- Return status
- User who submitted the task
- Messages triggered by the task

Returning Error Information from an Executable

To get information back to TaskMan from an executable, use any of these methods in the EXE:

- Put a message in the ProcessErrorLog table. These messages appear in the Task Messages tab on the Background Task History form.
- Print the error message in a file called

  taskman-install-directory\Output*task-name_task-number*.txt

  (for example, C:\Program Files\Infor*AppName*\Output\APChecks_435.txt) TaskMan will use this as an error message in the Background Task History.
- Return an integer error code. TaskMan puts the EXE return code in the BGTaskHistory table, with return code 0 indicating success.

Deleting Background Tasks

Deleting a WAITING Task

When the **Background Queue** form is used to create a job, it creates a record on the **Active Background Tasks** form with status WAITING and the appropriate task name and parameters. It then creates a SQL Server job with the name equal to *ActiveBackgroundTaskName_ActiveBackgroundTaskNumber_DatabaseName*. When you delete a task on the Active Background Tasks form, both the record and the corresponding SQL Server job are deleted.

To delete the task, mark the record for deletion on the **Active Background Tasks** form and then save your changes on the form.

Deleting a RUNNING Task

To delete a running task:

1. Find the task's process ID in the **Background Task History** form.
2. Log on to the utility server as an administrator and open the Windows Task Manager.
3. End the process ID you found in Step 1.

Row-Level Security

Using IDO Filters to Limit User Access

You can restrict the data that the user can see by placing filters on the IDO. Any smart client or web client that accesses the application can view an appropriate subset of the data in the collection, based on the filters applied to the IDO. IDO filters can be general (that is, applied to the IDO when anyone accesses it) or they can be constrained by user IDs or groups.

For example, you can create a filter so that salesperson users can only view their opportunities and not those of other salespeople. Or you can create a filter so that your customers who log into the application through a portal can see limited information about their orders.

Use the **Active** check box to turn filters on and off as needed, for example, during testing.

WARNING: By default, the **Active** check box is selected for all Infor-provided filters that are defined in the **Row Authorizations** form. If you turn off the Active check box for a filter, then any user who was previously restricted by the filter is no longer restricted. For example, if you turn off the Infor-provided filters that are set on the Interactions IDO, then any customer portal user can see ALL interactions in the system, not just the ones related to that customer.

Accessing IDO Filters

You can access the Row Authorizations form by any of these methods:

- From the Explorer, open the Row Authorizations form.
- From the Users form, click the **User IDO Filters** or **Group IDO Filters** button to open the form filtered on the selected user or group.
- From the Groups form, click the **User IDO Filters** or **Group IDO Filters** button to open the form filtered on the selected user or group.
- From the IDOs form, click the **IDO Filter** button to open the form filtered on the selected IDO.

Creating and Maintaining IDO Filters

To construct or maintain IDO filters from the **Row Authorizations** form:

NOTE: Remember that the drop-down list limits set in **View > Settings** are in effect. Use wildcards to display a subset of the items in a drop-down list, for example type **SLCu*** to list all the IDOs starting with "SLCu..."

1 In the IDO field, specify the IDO to be filtered. This populates the **Properties** drop-down list.

2 Optionally, specify either the User or the Group to restrict the filter.

3 Specify the Property that will be filtered.

4 Specify an Operator to use to compare the property with the value.

5 To specify a value, choose one of these options:

- Filter according to the current login, that is, the property is compared directly with the current user ID.
- Filter with a literal value, which you enter in the field next to this option. (Null is a valid literal value.)
- Filter by joining to a table that maps the User ID to some value(s). For example, this is useful for associating login IDs with other IDs such as customer number, salesperson ID, or vendor number.

6 If you chose to join to a table, select the mappings:

- Select a table from the drop-down list. This populates the two fields below it.
- Select the column containing user IDs.
- Select the column containing the value to be filtered against.

7 Click **Add This Clause**. The pseudo-SQL for the WHERE clause displays in the field at the bottom of the form.

8 To add more clauses to the filter, start again with step 1 and continue through step 6. If you want to set up this new clause as an OR Instead of AND with the Previous Clause, select that check box. Click **Add This Clause**.

9 When the IDO filter is complete, save the record.

10 When you are ready to activate the filter on the IDO, select Active and save the record again.

11 Whenever you create or change a filter for an IDO, you must unload metadata from the IDO Runtime on the utility server.

Examples

For several examples of how to set up IDO filters, see Examples: IDO Filters.

Notes

- When logged in as a super user, you always see unfiltered data.
- When you display a collection on a form, you can tell whether IDO filters have been applied by opening the **About This Form** dialog box. Any applied filters are listed in the dialog box.
- The IDO filter is an additional clause that is added to the pseudo-SQL WHERE clause which is transformed by the runtime into the SQL used to load the collection. The filters for an IDO are loaded with the rest of the metadata, so they are cached with the metadata objects. When the collection is loaded, the SQL command builder uses the metadata to add the IDO-level filter clauses to the WHERE clause, unless the load bypasses the filters (is loaded by a super user).

- The IDO filter clause is built by ANDing together all or any of these:
  - The general IDO filter, where both **User** and **Group** are blank
  - The user's IDO filter, where **User** equals the current login
  - The user's group's IDO filters, where **Group** equals any of the current user's group memberships. (The group filters are ORed together, since the user should be able to see the rows visible to any of the groups the user belongs to.)
- Placing many individual user filters on an IDO requires a lot of setup and maintenance time for the system administrator. Where possible, include multiple users in a group or cross-reference table, and then apply the filter to the group or the table join.
- Be aware that, when you use filters on IDOs that have multiple levels of IDO inheritance, or when you have users who are members of many groups that each have filters, the filter clauses can accumulate to the point where performance may suffer.
- Look at the pseudo-SQL query that is produced. If any of the columns in the query are not indexed, it can affect processing time.

Examples: IDO Filters

These examples apply to the Row Authorizations form.

Simple Example: Username Property = Current User

You want to allow all users to see their own user information on the Users form (description, e-mail address, group memberships, etc.) but not any other user's information. To do this, you would set the fields as follows:

Field	Setting	Notes
IDO	UserNames	The filter applies only to the UserNames IDO.
User		Leave blank so the filter applies to all users.
Group		Leave blank so the filter applies to users in all groups.
Property	Username	the User ID field on the Users form is bound to the UserNames.Username property. (The Usernames.Userid property is only used internally.)
(Operator)	=	
This	Current	

Value	User

When you click **Add This Clause**, the pseudo-SQL query that displays looks like this:

```
(Username = dbo.UserNameSp())
```

After you activate and save the filter, any user who logs in (and who has the proper authorizations for the form) can open the Users form but will see only the record that matches their user ID.

Transaction Isolation Levels

About Transaction Isolation Levels

The Collection Read Mode setting corresponds to the Transact-SQL statement SET TRANSACTION ISOLATION LEVEL READ COMMITTED [*or* UNCOMMITTED]. It applies to queries that load primary collections, secondary collections, and lists, and to "in-collection" validations. It also applies to background-task queries that generate reports and to background-task stored procedures.

The base, systemwide transaction isolation level is set on the **Process Defaults** form. You can override this setting (UNCOMMITTED or COMMITTED) for individual reports and stored procedures on the **Background Task Definitions** form. You can also override the setting at the form level through the **Read Mode** field in design mode.

The UNCOMMITTED setting allows the reading of uncommitted data. Users do not have to wait for other long-running transactions accessing the same dataset to complete before their queries can complete. However, an uncommitted record may be rolled back between the time the query displays it and the time the user attempts to save updates to it. Users cannot save a record if uncommitted data were rolled back at some time after the query, because the optimistic locking will fail. The user must refresh the record to get valid data before making changes and saving the data.

With the COMMITTED setting, a query reads committed data and returns only data for which the query can get a shared lock.

Setting Transaction Isolation Levels

To change the base transaction isolation level

1. Open the Process Defaults form.
2. Select the Collection Read Mode default.
3. In the Default Value field, type either COMMITTED or UNCOMMITTED. Note that if the field is blank, queries use COMMITTED as the transaction isolation level.

To change the transaction isolation level for a report or stored procedure

1. Open the Background Task Definitions form.
2. Select a background task of the type RPT or SP.
3. In the Isolation Level field, select Read Committed or Read Uncommitted. Note that if the field is blank, queries use the transaction isolation level (collection read mode) set on the Process Defaults form.

To change the transaction isolation level for a collection in a form

See the online documentation for WinStudio design mode.

Index

G

I

L

M

O

P

R

S

T

U